Authoring and Formatting of Hypermedia Documents

Luiz Fernando G. Soares
lfgs@inf.puc-rio.br

Débora Muchaluat
debora@telemidia.puc-rio.br

Rogério F. Rodrigues
rogerio@telemidia.puc-rio.br

Depto. de Informática, PUC-Rio
R. Marquês de São Vicente 225
22453-900 - Rio de Janeiro, Brasil

Abstract

In spite of the number of multimedia and hypermedia authoring and formatting related works, and in spite of research advances already obtained, a long and winding road still remains to be trek. This paper considers some issues on the development of multimedia and hypermedia authoring and formatting tools, examining the current state of the art. Moreover, it discusses several research challenges that need to be addressed. The paper stresses the importance of document logical structuring, and considers the use of compositions in its support. Integration of open hypermedia systems to the WWW is also examined.

1. Introduction

Hypermedia systems must satisfy, at least, three main requirements. First, they must support different types of media segments. Second, they must allow explicit definition of temporal and spatial relationships among several media segments, including those relations triggered by user interaction. Third, they must implement a versatile spatio-temporal formatting algorithm.

Moreover, hypermedia systems may introduce several desirable facilities, such as:

- allowing a structured authoring (hierarchical or not) of documents;
- supporting, for each media segment, a rich set of capabilities, like: presentation duration flexibility, different exhibition alternatives, behavior changes during presentation, etc.;
- supporting, for each media segment, relations anchoring on internal points of the segment (fine granularity), and not only on its start or end points (coarse granularity);
- allowing quality of service (QoS) definition required by media segments (e.g., jitters, bandwidth, etc.);
- allowing a presentation environment description (e.g., network delay, supported devices, etc.);
- allowing the explicit definition of temporal relationships with non deterministic times;
- supporting a temporal formatting algorithm that considers the non determinism and allows correcting the presentation on-the-fly, when unpredictable events occur (e.g., network delay, user interaction, etc.); and
- supporting a temporal formatting algorithm that takes profit of media segments' QoS specification and environment characteristics, for instance, realizing pre-fetch of objects' content.

Hypermedia systems have been addressed in three different levels in the literature: storage, specification (authoring) and execution (formatting). This paper mainly focuses on the latter two. The specification level aims at defining hypermedia application requirements and associated constraints. The formatting level refers to the development of protocols and schemes for document exhibition, dealing with intra-media temporal synchronization, and inter-media temporal and spatial synchronization. Several systems discussed in the literature deals with authoring and formatting issues.

In this paper, we consider the development of multimedia and hypermedia authoring and formatting tools, examining the current state of the art. Moreover, we discuss a set of research challenges that need to be addressed before the full potential of multimedia output technology can effectively be utilized to share information. The paper is organized as follows. In Section 2, issues related to hypermedia document authoring are discussed, stressing the importance of document logical structuring, and showing how to use compositions to support that structuring. Section 3 addresses the problem of temporal and spatial formatting. The integration of open hypermedia systems with the WWW is discussed in Section 4. Section 5 contains our final remarks.

2. Structured Authoring of Hypermedia Documents With Temporal Constraints

Authoring tools are based on sophisticated conceptual models which are rich in their semantic capabilities to represent complex multimedia objects and express their relationships requirements. Before beginning a discussion about structured authoring let us define some terms used with different meaning in the literature. We will follow as much as possible the definitions used in Pérez-Luque and Little Temporal Reference Framework [PeLi96].

An authoring environment should offer good editing and browsing tools for defining the logical structure of a document, its components' content and the content granularity, which specifies the set of information units that can be marked and used in the definition of events. The exact notion of information unit and marked information unit (an *anchor*) is part of the definition of the document's component, from here on called a *node*. For example, an information unit of a video node could be a frame, while an information unit of a text node could be a word or a character. Any subset of information units of a node may be marked.

An *event* is an occurrence in time that can be instantaneous or can occur over some time period. A *presentation event* is defined by the presentation of a marked set of information units of a node. A *selection event* is defined by the selection of a marked set of information units of a node. An *attribution event* is defined by the changing of an attribute of a node.

Finding a general way to indicate the presence of an anchor in dynamic data, or providing means of "clicking" a portion of video and audio, still remain difficult and unsolved problem. SMIL [W3C98] allows anchoring in an entire dynamic object, or even in spatial or temporal subparts of an object, defined by their corresponding spatial coordinates or temporal instant, respectively. However, the problem of allowing a motion anchor in a video, for example, remains unsolved [WBHT97]. In order to give an idea of this need, imagine an airplane in a video flying over a country that displays a text information about the population of the corresponding state which it is over.

Another open issue regards the creation of an object and its anchors from a particular information site. The development of *adaptative media objects*, as called by Bulterman and Hardman [BuHa95], supports reuse and tailoring of data items. Three aspects of object integration into documents need to be supported for adaptative media object creation: locating a particular object from an object store, extracting the relevant portion of the object for use in a particular document and transforming the representation of the fragment to meet the dynamic needs of the runtime presentation environment. Some related works were done in the area of database systems, but they are just at the beginning.

Authoring conceptual models must also be rich in their semantic capabilities to express relationships among the document's components (nodes). There are several types of relations that must be supported, among others:

- reference relations: for example those leading to a small note or those that jump to an entirely new section;
- context relations: for example those that specify a hierarchical structure of a document, such as a book and its chapters, these chapters and their sections, and so on;
- synchronization relations: those that define both temporal and spatial ordering of objects;
- derivation relations: for example those that bind a data with the object from which it was created; and
- task relations: those that organize tasks in a cooperative work.

The notion of structured documents arises from the introduction of the composition concept, as a container of documents' components and their relationships. However, the logical structure depends on which relations we are interested in, as it is discussed in the next section.

2.1 - Compositions and Structured Documents

The structured definition of documents is desirable as it carries built-in concepts of modularity, encapsulation and abstraction. Structure-based authoring requires a well-developed model for hypermedia documents. All these models separate the definition of the logical structure of a document and its associated media objects. The composition concept is introduced as a container of documents' components (nodes) — media objects or even compositions, recursively — and possibly their relationships, allowing support for both top-down and bottom-up design. From the logical structure some relationships may be derived, being the others defined by other model entities.

There are several structure-based authoring models depending on which type of the above mentioned relationships they are interested in structuring.

- **Composition for context structuring**

This is the composition usually defined in structured hypertext models. It is found in several models, for example, in the *context nodes* of [CTRS91] and [WiLe97], where the context relations are implicitly derived from compositions. Figure 1 shows an example of a book (composition B1) composed by chapters C1, C2 and C3. Composition C1 (chapter C1) contains the sections S1.1 and S1.2 and a media object O1, etc. Other relations, for example reference relations, may be represented by links. For instance, a reference relationship between the media object O1 and chapter C3 is represented by the link entity l1 in the Figure 1.

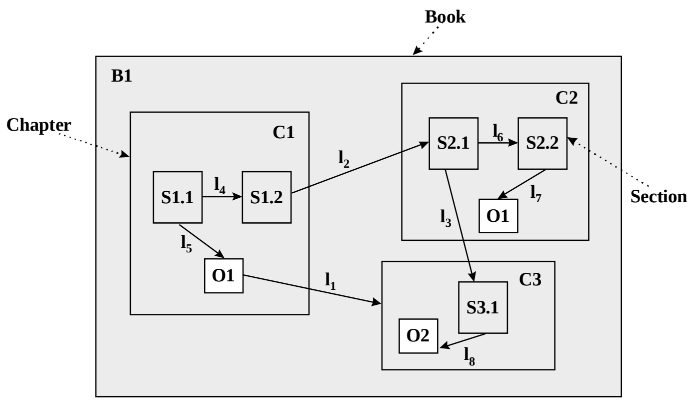


Figure 1 - Context structuring

- **Composition for spatio-temporal presentation structuring**

In this case, as stated in [VaTS97], composition is used to represent both temporal and spatial ordering of its objects. In other words, it is an entity that embeds a set of nodes with their spatial and temporal presentation relations. These compositions are found in several conceptual models, for example those of CMIF [RJMB93], SMIL [W3C98], I-HTSPN [WSSD96] and in the MOAP and IMD models of [VaMo93, Vazi96 and VSTH98].

Explicit temporal relations between nodes are defined in other entities, the *sync arcs* of CMIF, the Petri Net's transitions of I-HTSPN and the *scenario_tuples* (and *actions*) of MOAP and IMD.

Temporal relations among nodes can also be derived implicitly by the composition type, as is the case of CMIF and SMIL. In CMIF composition, for example, relationships are given implicitly by the composition type, that can be *parallel* or *sequential*, where all components must be presented in parallel or in sequence, respectively. The experiment warns against the use of too many links (in its case s*ync arcs*) which, while bringing the advantages of more flexible access structures than the implicit relations given by compositions, have to be weighted against the cost of the extra complexity.

Usually, in models where compositions structure temporal presentations, context relations as well as reference relations are given by links. There is no way to distinguish the semantic of these two types of relation through that unique model entity. For example, in [VaMo93] *button_lists* define context relations. This was one of the problems raised by Halasz in his seven issues on Notecards [Hala88].

- **Composition for grouping correlated versions of nodes**

Here composition is used to group objects in order to maintain and manipulate a history of changes in these objects, that is, it groups nodes that represent versions of the same entity, at some level of abstraction. The grouped versions can even be the previously mentioned type of compositions (for context and presentation relations), if the models permit them, allowing to explore and manage several alternate configurations for a network of nodes (related by context or presentation purposes). For example, in Software Engineering there are two levels of versioning. The lowest level corresponds to different modules that make up programs. Naturally, all versions of a module may be grouped in a version composition. The other level is the configuration, that is, the description of which modules the program is made from, and how the modules should be put together to compose the program. A configuration can be modeled by a context or presentation composition. Configurations can also be interpreted as versions of the same object and grouped together into a version composition.

Compositions for version control can be found in the *mobs* of CoVer [Haak92], *version groups* of HyperPro [Oste92] and *version contexts* of [SoCR94].

- **Composition for task maintenance**

Conceptual hypermedia models that allow cooperative work usually use compositions to structure cooperative tasks that compound the cooperative environment. Examples of such compositions are the *tasks* of CoVer [Haak92] and the *private bases* of [SoCR94]. In both models tasks are a set of interrelated document's components and sub-tasks, recursively.

Cooperative work is frequently associated with version control, notification mechanisms and a powerful query language, as will be briefly discussed later.

- **Composition as a class from which other composition types can be derived as subclasses**

This is the case of compositions defined in models that have more than one structuring purpose, or in meta models for hypermedia authoring conceptual models. Examples of these compositions can be found in the already mentioned CoVer and HyperPro but specially in NCM [SoCR95].

Figure 2 shows the NCM class hierarchy, where the nodes in gray background are those that are subject to version mechanisms [SoCR95]. In the HyperProp system, one implementation of NCM, *user contexts* are used for context structuring; *version context* for version structuring; *private base*, *public hyperbase* and *annotation* for cooperative work structuring; and *trail* for guided tours structuring. In HyperProp, temporal and spatial relationships are defined by links.

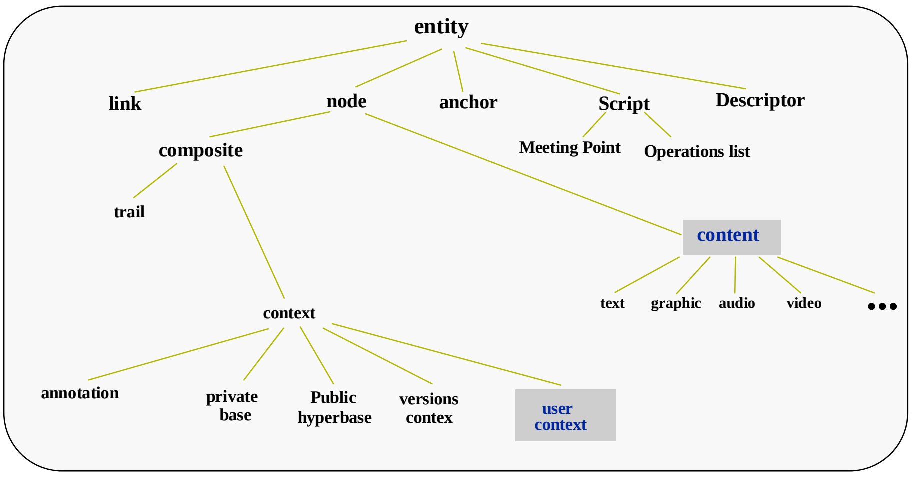


Figure 2 - NCM (Nested Context Model) class hierarchy

It is worth to note that the composition class of NCM can also be specialized in parallel composition and sequential composition subclasses, with the same functionality defined in CMIF, although this is not yet implemented in the current version of HyperProp.

In spite of the five types of compositions composition used, compositions can be used to group only structured related nodes or to group nodes and their relations. In the first case, relations represented by links must be stored in an independent repository which sometimes is unique, as is the case of Hyper-G [AnKM95], CMIF and DHM [GrBS97]. In the second case, compositions group nodes and links, as is the case of AHM [HaBR94], where composite nodes contain *sync arcs*, and the case of NCM, where composite nodes are defined as a collection of nodes and links among the nodes, recursively contained in the composite node. Note that when links are defined in the composite node we can have a structured reuse not only of relations implicitly given by the composition but also of relations defined by links in the composition.

Compositions have methods to be played, stopped, etc. But what is playing a composition? It depends on the relation it represents. For example, playing a composition representing only spatial and temporal relations is to start the sequence of its components presentation; playing a composition representing context relation is to display its context structure, for example in a map similar to that of Figure 1.

Despite of several works discussing compositions in their different types of use, several open issues still remain. Many systems enable the creation of the document structure but give an author no support in determining what would be the most effective structure to be created. Support for automatic generation of document structure is in the beginning, for example, automatic composition extraction from spatio-temporal specifications. Even a more difficult issue is the structured document automatic generation. For instance, based on a subject and the knowledge of a user's task a structured document can be generated by combining appropriate media items into coherent groups with derived temporal and layout presentation specifications. The goal in this case is to (semi-)automatically generate presentations, thus reducing the effort required by the publisher to cater for such a variety of users. Some of this works are reported in [HaBu95 and HaWB98].

2.2 - Authoring Models

Conceptual models differ in their goals, in what relations they allow and how these relations are defined.

Authoring models for hypertext documents place emphasis on the reference relation navigation. Usually they only use composition to allow a hierarchical definition of the context structure of documents.

Authoring models for multimedia presentations place emphasis on time. Usually they only use compositions to allow a hierarchical definition of presentations. As sometimes synchronization constraints are defined among items that are contextual structurally related more often than items that are farther apart in the document, structure compositions also give the context structure of the document. However, this cannot be generalized. For example, in the Figure 1, assume that the component *S1.2* of the composition modeling Chapter *C1* is the navigation source to the *S2.1* component of the composition modeling Chapter *C2*, for instance specifying that *S2.1* must start after the end of *S1.2* (link *l2*). If both components are defined in the same composition, then the context structure (chapters, sections, etc.) will be lost.

Authoring models for cooperative work usually place emphasis in hierarchical structuring of cooperative tasks, but also need to group versions and contextually related nodes in compositions.

We can have a different authoring tool for each purpose but we can also have an integrated tool, preferably based on a general purpose hypermedia conceptual model. This is one of HyperProp's goals. Whether this will be efficient or not is still a question to be answered.

A general purpose hypermedia conceptual model allows not only the implementation of an integrated authoring tool, but can also be used as a meta model to derive efficient specific authoring tools through the appropriate specialization of its entities.

Authoring models must deal with several other issues besides structuring. Some of them are discussed in the remaining subsections.

2.3 - Spatial and Temporal Synchronization

There are several temporal hypermedia and multimedia models proposed in the literature based on various paradigms. Pérez-Luque and Little in [PeLi96] discuss several models and respective paradigms based on their proposed Temporal Reference Framework for Multimedia Synchronization. Based on this framework, models are classified and their equivalence are discussed for specific as well as general scenarios.

Almost all model proposals begin or end comparing themselves with other models (commercials or academics). We will have a different approach here. Instead of discussing models or model's paradigms we will stress some points we consider important to be satisfied in order to highlight some smart solutions and identify some open issues. Characteristics in some way satisfied by all models are not discussed.

- **Spatial relationships**

Spatial synchronization can be defined, optionally, in the same relationships used to define temporal synchronization. A link synchronization relation, for example, cannot only specify the moment that a given presentation event will happen but also how the presentation inside the current scenario must occur. The *scenario_tuples* (and *actions*) of MOAP and IMD are examples. Optionally, spatial synchronization can be defined in other entity. This is the case of Firefly [BuZe93] and NCM, where behavior changes and spatial synchronization are handled by operation lists, CMIF, where they are defined in the *channels*, SMIL, where they are defined in the *layout* element, and I-HTSPN, where they are defined in the *presentation specification* object. In all these models, except MOAP and IMD, the spatial synchronization is established by the placement of objects in an absolute spatial coordinate system, based on a paradigm analogous to the timeline for temporal synchronization, with all the disadvantages reported on several works in the literature.

The spatial synchronization of IMD [VaTS98] is richer than those previous models, as it allows event constraint based spatial relationships, generalizing the constraint instant-based and interval-based paradigm for temporal relationships, present in all mentioned models.

- **Relations based on event states**

Usually we need to test platform dependent activities in order to start the presentation of an object. For example, we may need to test if a video node was got from a remote server before initiating its presentation in parallel with a local audio.

In NCM an event can be in one of the following states: *sleeping*, *preparing*, *prepared*, *occurring* and *paused*. Moreover, every event has an associated attribute, named *occurred*, which counts how many times an event transits from occurring to prepared state during a document presentation.

Intuitively, taking a presentation event as an example (see Figure 3), it starts in the sleeping state. It goes to the preparing state while some pre-fetch procedure of its information units is being executed. At the end of the procedure, the event goes to the prepared state. At the beginning of the information units exhibition it goes to the occurring state. If the exhibition is temporarily suspended, the event stays in the paused state, while the situation lasts. At the end of the exhibition, the event comes back to the prepared state, when the attribute occurred is incremented. Obviously, instantaneous events, like selection and attribution, stay in the occurring state only during an infinitesimal time. The event state machine in HyperProp is executed under the document formatter responsibility, as will be discussed in Section 3.

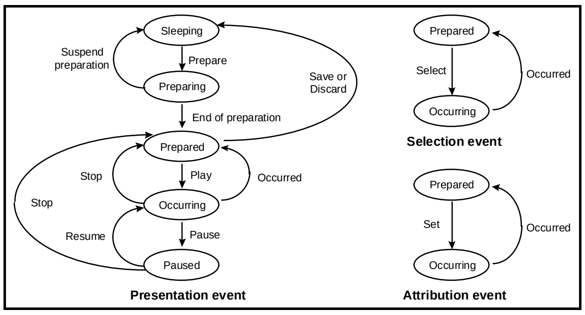


Figure 3 - HyperProp (NCM) events state machine

- **Indefinite relationships**

As defined by [PeLi96], indefinite spatio-temporal relationships are those temporal relations between time instants or time intervals that are not explicitly or unambiguously given. Usually they are expressed as disjunctions of the basic spatio-temporal relationships (for example, “at the same time or after”). Several models, exemplified by NCM and IMD, present some support to indefinite relationships. In [PeLi96 and VaTS98] all possible indefinite relations for instant and interval based spatio-temporal relationships are discussed. If all those relations are needed is an issue to be tested.

- **Multipoint relations and contextual links**

Although relations are usually directional (usually expressed by a link) they can be followed in both directions in almost all conceptual models implementations. However, few models allow n:m relationships, that is, relations with several source and destination end points that can test conditions on the source end point set in order to trigger actions on the destination end point set. I-HTSPN and NCM present a general solution for the specification of these relationships.

Multipoint relations are also an elegant solution to the problem of defining a context for a link, stated by [HaBR93]: “In hypermedia it is useful for the author, however, to be able to specify which parts of the presentation should remain and which should be replaced when a link is followed. A source context for a link is that part of a hypermedia presentation affected by initiating a link, and a destination context is that part of the presentation which is played on arriving at the destination of a link”. Context for a link is a natural consequence of a multipoint relation and not a new attribute for a 1:1 relation.

- **Relations based on combinations of presentation and selection events**

Some models (CMIF, AHM, etc.) define temporal and spatial relations only between presentation events; relation between selection events are defined in another model entity (usually a link) completely separate from spatio-temporal relationships. Some models even argue that a hypermedia model needs to express time-based relations, but these should be treated as presentation information, and kept separate from the link-based structure information. In these models it is not possible to merge conditions spatio-time dependent and user interaction dependent to trigger a navigation. For example, the situation “play an audio explanation when a user selects a text button during a video exhibition” cannot be represented. Note that this is a typical case of n:m relationship. This situation is not rare and models should support them. Examples of solutions can be found in IMD, NCM and I-HTSPN.

- **Compositions with implicit time relations**

As already mentioned in the previous subsection, this can prove to be very useful for an author. CMIF has as one of its goals to provide an authoring system which supports the author as much as possible in thinking in high-level terms, and which automatically generates the corresponding high level timing and placement information. Several CMIF’s utilization reports state the great utility of this facility.

- **Relations between nodes inside different compositions.**
  Relation inheritance in compositions.

Relation inheritance in compositions means that relations among nodes can be defined in any composition that recursively contains these nodes.

In several models relations represented by links (spatio-temporal or only reference relation) must be stored in an independent repository. As mentioned before, this is the case of Hyper-G, CMIF and DHM. Relation inheritance, in this case, does not make sense.

In other models compositions group nodes and links, but the relation is stored in an ancestor of both nodes which is lowest in the hierarchy, as is the case of the *sync arcs* of AHM. Again, relation inheritance does not make sense in this case.

In several other models (MOAP and I-HTSPN, for example), relations are defined between events contained in a composition. They allow a composition as an end point of a relation, but not a component inside a composition. This will prevent relation inheritance.

Note that when relations are defined in the composite node we can have a structured reuse of relations implicitly given by the composition as well as of relations explicitly defined inside the composition. Indeed, relation inheritance in composition nesting (as exemplified in NCM) is very important in order to allow a more general composition (and thus structure) reuse. As an example, suppose the composition representing the book chapter C1 of Figure 1. For a book given to a reader (composition *B_2*), it could be desirable to introduce a relation between sections S1.1 and S1.2 of C1, to give a hint of related matters; for another advanced reader, the book (composition *B_1*) should be delivered without that relation, as in Figure 1. Note that *B_2* could be defined as a composition containing *B_1* and the introduced relation, reusing all the structure of *B_1* as shown in Figure 4.

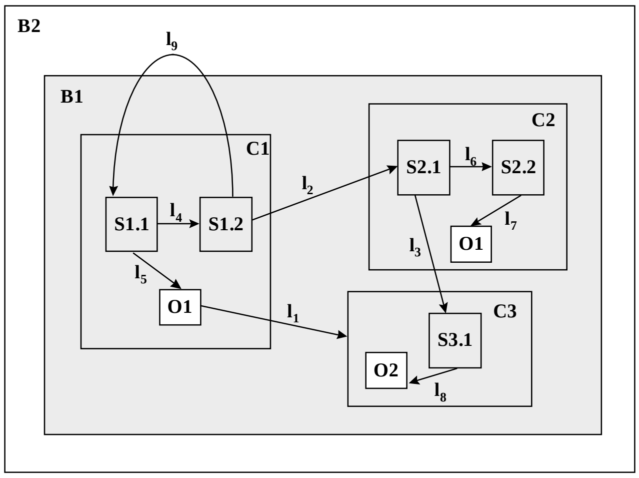


Figure 4 - Relation inheritance

- **Compositions with different entry points**

Different entry points in a composition is desirable since it permits different synchronization presentations of the composition's node components. This is also an important characteristic to allow structure reuse. NCM is an example of a model that allows such facility.

- **Layout definition independent of a node content**

Several authoring systems (Firefly, CMIF, I-HTSPN, NCM, SMIL, IMD, etc.) defines the layout of their components presentation separated from the associated data object. Again, this will allow a better reuse of objects. For example, using distinct layouts, one can define different presentations for the same media object. A text media segment can be presented as text, or it can be synthesized as audio, depending on the layout. Note that as different layouts can lead to different event duration, different quality of presentation and different platform requirements, the definition of all these issues should also be part of the layout and not part of the media object.

The separation of the layout from the media object will also allow a layout specification made by the reader, as will be detailed in the next subsection.

2.4 Temporal Behavior Specification

An authoring environment should permit the definition of each component expected behavior when presented. As mentioned in the previous section, such specification should be made apart from the media object to be presented. This is the case of Firefly and NCM, where behavioral changes and spatial synchronization are handled by *operation lists* associated with media objects (in NCM these lists are contained in an entity called *descriptor*).

Some systems allow a join definition of the spatial synchronization, specifying it in an entity that can be shared by several media objects. This is the case of CMIF, where an event is assigned to a *channel*, which is an abstraction of a set of properties shared by other objects of the same media type. The use of CMIF channels can be very desirable. For example, it ensures that the system's audio driver will not need to be reinitialized for every data block sent to the device. It also allows special effects to be introduced when changing one object presentation to another, what would be difficult to get with individual layout entities.

Whatsoever is the way the presentation behavior is specified, two issues are very important to be handled. These issues still need better proposals that have their usefulness effectively tested. One is the already mentioned spatial synchronization between objects based on constraints and not on absolute values of some coordinate space (for example, spatial relationships between channels). Reference [VaTS96] has good contributions on this matter. The other issue is to allow behavior changes, as identified in Firefly's operation lists. Both Firefly and NCM only treat discrete behavior changes. That is, they identify time instants where behavior changes must be made, but do not allow continuous behavior changes. In [NaKa97] good ideas about the matter are given.

It is also important to have alternative layouts for the same media object to allow platform QoS adaptation, user choice and different presentations for a document depending on the navigation start point. Alternative layouts are allowed by several systems and models like NCM and SMIL. It should be pointed out that providing truly adaptative documents, that match their performance and appearance characteristics to the available resources is a fascinating research topic.

The presentation of an object is created by the spatio-temporal formatter from its corresponding layout specification and data. Given a media object, an associated layout can be specified on-the-fly by the end user, or in the author's specification. Presentation choices only in the document specification give the author the responsibility for designing and implementing alternate presentations and (s)he often does not know the diverse needs of users. An authoring model should allow that the layout may be defined in a media object attribute, or in a link (or any other entity specifying spatio-temporal, and reference relation) that has the event as its destination anchor. Composite nodes may also have, for each node they contain, an attribute that can be used to store its layout identifier.

In NCM, when presenting a node, the descriptor (layout specification) explicitly defined on-the-fly by the end user bypasses the descriptors defined during the authoring phase. These in turn have the following precedence order: first, that defined in the link used to reach the node; second, that defined in the composite node that contains the node, if it is the case; third, that defined within the node; and finally, the default descriptor defined in the node class.

Style Sheets, a specification of the appearance for a collection of documents with similar structure, is now receiving a lot of research attention in the specification of multimedia documents. Style sheets and media object layout definition (one can see this layout as a small style sheet for just one object document) can be defined using a GUI, or specification languages such as CSS [LiBO96], DSSSL [ISO96], or PSL [Muns96].

Declarative document presentation specification, such as SMIL, may import layout specifications defined in different languages. Layout specification languages is another current research topic of great interest.

2.5 - Versioning and Cooperative Work

Even though the need for version control in hypermedia systems has long been recognized, the complexity of the interaction between version control and other requirements has apparently delayed the work in the area. In the years 1992 to 1994, some proposals were made; among them the HyperPro [Oste92], CoVer [Haak92] and HyperProp [CTRS91 and SoCR94] systems must be mentioned, what led up to the "Workshop on Versioning in Hypertext System", held together with ECHT'94. After then, few works appeared, besides those related to the already mentioned CoVer [Haak94 and Haak96] and HyperProp [SoCR95 and Soar98]. Cooperative work has mainly been treated in the works of CoVer.

A good discussion on version control can be found in [SoCR95] and [Haak96], that also makes a good discussion on cooperative authoring. Here we will only stress some authoring model requirements that come from these areas and some still open issues on them.

- **Same node inclusion in different compositions**

Authoring tools should allow the reuse of part of a document in the creation of a larger, more complex hypermedia document. Several models allow the reuse of data once their media objects only reference the data files that can be shared. However, it is also important to reuse structures. Suppose, for example, that we want to reuse the composite node *S1.1* of Figure 4 inside composite node *C2*. We could, for example, copy all the structure into *C2* and then redefine all nodes identifiers. This would also imply in redefining all relations end points recursively defined in the new *S1.1'*. Moreover, the information that *S1.1* and *S1.1'* were the same structure and that changes in one would need to reflect in the other must be stored somewhere. A more general solution would be to allow a node to be included in more than one composition. This would have some consequences in identifying a node, as will be discussed in the next item.

- **Perspective of a node**
  Redefining end points of relations (links)

If models allow different composite nodes to contain the same node and composite nodes to be nested to any depth, it is necessary to introduce the concept of perspective. Intuitively, the perspective of a node identifies through which sequence of nested composite nodes a given node instance is being observed. Formally, a *perspective* of a node N is a sequence $P=(N_m,...,N_1)$, with $m \geq$### 1, such that $N_1=N$, N_{i+1} is a composite node, N_i is contained in N_{i+1}, for $i \in [1,m)$ and N_m is not contained in any node. Note that there can be several different perspectives for the same node N, if this node is contained in more than one composite node.

The node end point of a relation (a link for example) will have to be identified by the sequence $<(N_k,...,N_1 >$ such that N_1 is a node, N_{i+1} is a composite node and N_i is contained in N_{i+1}, for all $i \in [1,k)$, with $k > 0$. The node N_1 is called an *anchor node*. The node N_k is called a *base node* of the relation, and must be contained in the composite node that contains the relation.

- **Version propagation**

In any system with composite nodes, one may ask what happens to a composite node when a new version of one of its components is created. A system is said to offer *automatic version propagation* when new versions of the composite nodes that contain a node *N* are automatically created each time a new version of *N* is created. If a node can be contained in many different composite nodes version propagation may cause the creation of a large number of often undesirable nodes.

NCM provides mechanisms to avoid the proliferation of useless versions, based on the concepts of *private base* and *node state*, on a version propagation mechanism and on different primitives (instead of only the usually known *check-in*) to create versions. However, the matter demands more work and is far from being resolved.

- **Different presentation as versions, immutable attributes and link versioning**

Although the literature stresses the importance of considering representation objects (aggregation of objects with their layout specifications) derived from the same object as different versions, few proposals discuss that possibility. Indeed, most works that mention that issue allow only one representation version of an object, by simplicity. NCM allows that distinct presentations (representations) of the same piece of information (in the same or different media) be treated as versions of that piece of information. This extended use of the notion of version, coupled with a notification mechanism, provides a good basis for cooperative work. However, different representation versions of the same object can generate a proliferation of useless data versions and make it very hard for the system to guarantee version history consistency, if well-defined rules are not established.

In hypermedia it might be too simplistic to have versions of nodes to be completely immutable, that is any change in any attribute of a node implying in a new version. It is not even obvious that the content data attribute of a version should be always immutable. Several models (NCM, HyperPro and CoVer) allow each attribute (including data content) to be specified as *versionable* or *non-versionable*. The value of a non-versionable attribute may be modified without creating a new version. Modifications in versionable attribute values have to be made on a new version of the object. Of course, some kind of notification mechanism will be needed to enhance version support, specially in the case of concurrent update of non-versionable attributes. Moreover, in NCM the user may specify if the addition of new attributes to a node is allowed without creating a new version.

A few models, like CoVer, support link versioning, although it is not clear in any of them how versioned links are handled. As stated in [Soar98], when we allow several representation versions to be derived from the same object, link versions proliferation seems to be a more complex problem than version propagation of nodes.

Indeed, much work must be done regarding version control, none of the above mentioned facilities are proved to be necessary. Few solutions were proposed, only some of them were implemented, and almost none of them was rigorously tested.

- **Cooperative work and notification mechanism**

To support cooperative work, an authoring environment must naturally allow users to share information. However, the environment must also provide some form of private information, for security reasons as well as to allow fragmentation of the hyperbase into smaller units in order to reduce the navigation space.

Few systems provide support for cooperative work. CoVer defines a hierarchical task model for cooperative authoring, based on a powerful model. However, the matter demands more work and is far from being resolved. Version control, cooperative work and notification mechanisms are related matters that must be treated together and that demands much more research attention.

2.6 - Turning Back to Compositions

Briefly, the use of composite nodes in the modeling of structured documents (speaking about structuring in its *broad sense*) must bring out some properties, such as:

- composition nesting, that is, compositions that contain other compositions;
- grouping of the components of a document and the relationships among them independent of their types (synchronization relationships for presentation, reference relationships for usual hyperlinks navigation, derivation relationships, etc.);
- pertinence of the same node in different composite nodes, permitting this node to have different behavior depending on the composition it is in (for example, in the Figure 1 the node *O1* is contained in composite nodes *C1* and *C2.* In *C1*, *O1* takes a part in two relations represented by links *l1* and *l5,* while in composition *C2*, *O1* only takes a part in the relation represented by link *l7*);
- composite nodes use as a new type of node, in all senses, that is:
    - ∗ That they can have its structure presented[1] — since in a presentation, it is important to exhibit not only the data content of a document, but also its structure specified in the composite node (for example, when accessing a book chapter modeled as a composite node, besides seeing its content, one may want to visualize its section structuring).
    - ∗ that different entry points (anchors) in a composition can be defined, i. e., that in a composition, components may have different presentations, depending on the entry point. As a consequence we can have different sequence of composition's components presentation depending on the entry point. Thus, the duration of a composition (duration of its components exhibition) will depend not only on the duration of its components, but also on the associated entry point;
    - ∗ that relations among compositions can be defined.
- inheritance in the composition nesting, in the sense that relations may be defined in a composition *C*, referencing components recursively contained in *C*. This mechanism is extremely important in composition reusing, as stated in Section 2.3.
- composition specialization to implicitly define any form of synchronization, derivation, or contextual relationship.

The object oriented model NCM defines compositions with all the requirements specified before. The HyperProp system does not implement however a composition sub-class that implicitly defines temporal synchronization, as in SMIL and CMIF, a very useful mechanism as stated in those works.

[1] Composite node presentation is different from the presentation of its components. Composite node presentation is the exhibition of the structure defined in the composition and not the exhibition of each one of its components.

2.7 - Graphical Interfaces for Authoring

An author (who is usually a non-technical person) needs tools for high level but complete description of all aspects of a multimedia application. Several systems (CMIFed, HyperProp, I-HTSPN, etc.) offer authoring tools with graphical interfaces, making easier the task of document authoring. A graphical authoring environment should support both top-down and bottom-up construction. Constraints among media items (events) should be preferably defined directly among those items.

Graphical authoring tools are usually based on different views that allow the specification of all types of relationships defined previously. Each view favor one type of relation definition and all views must work in an integrated way.

The first view, common in all systems, called Structural View in HyperProp (Hierarchy View in CMIFed jargon), supports browsing and editing the logical structure of hyperdocuments. As ever, what is defined as "logical structure" depends on the conceptual model in use, as mentioned in Section 2.2. This view must allow, at least, grouping nodes into compositions but, as will be discussed, it is also a good place to group relations represented by links into compositions. Briefly, the Structural view must provide features for editing nodes and links and grouping them into compositions, as illustrated in the upper plane of Figure 5. In this view, nodes are represented by rectangles, links are represented by lines and the containment relationship of composite nodes is represented by the inclusion of rectangles (nodes) and lines (links), in another rectangle (composite node).

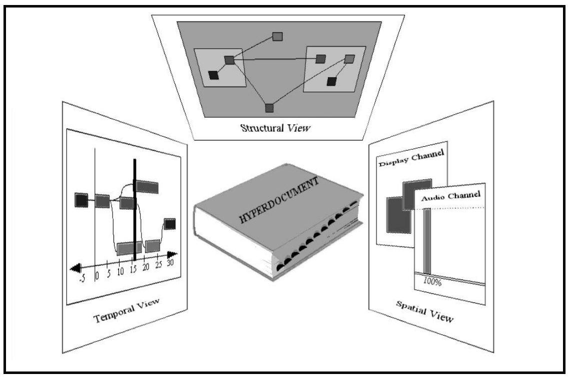


Figure 5 - Different views of a hypermedia document

CMIFed hierarchy view is used only for viewing the tree of nested presentations, and for viewing the tree of events within an atomic presentation. It does not show nor allow link definition. This is a consequence of the fact that in CMIF links (including CMIF *sync arcs*) do not pertain to compositions, but are stored in a separate repository.

In Time Petri Net based models like I-HTSPN, an event is represented by a place (a node in a graph drawn by the tool) and by associating a tuple [t_{min}, t_{opt}, t_{max}] to an outgoing arc from the place. The tuple [t_{min}, t_{max}] specifies the possible duration interval of the event, and t_{opt} specifies the time duration with best QoS presentation. Delays can be introduced in I-HTSPN through adding places with outgoing arcs associated with the possible interval of delay occurrence. A hyperlink (reference relation) is represented by a new place type having an usual tuple [t_{min}, *, t_{max}] associated to an outgoing arc, representing the possible selection interval of the link. Compositions are represented by a new type of place in the Petri network. A composite place and its related subnet must not only be structurally equivalent (i.e., the subnet must have an input place and an output place), but also temporally equivalent. Compositions can be used as an end point of a link, but we cannot have a component inside a composition as an end point.

Usually, the Structural View has a very complex structure, requiring sophisticated algorithms to build the view. The main motivation is to balance local detail and global context with respect to a node in focus. Local detail is necessary to give information about the navigation possibilities, depending on the focus in the document structure. Global context is important to give information about the focus position within the overall structure.

There are several techniques to display graph structures [SaBr94]:

- Presenting all nodes and links of the graph, which allows users to have a global view of the structure. The drawback is that maps become confusing and do not help user navigation when the number of nodes and links increases.
- Using scroll and zoom to view portions of the graph, which makes readable maps, but loses the overall structure.
- Using two or more views, one with a global view and another with a small zoomed portion of the graph, which has the advantage of displaying local details and the overall structure of the information space, but forces the user to mentally integrate the views.
- Using filtering mechanisms, such as fisheye views, to build a global view, that only displays what is more interesting to the user at a certain moment, maintaining map legibility. The difficulty is to find out *what* is more interesting to the user at that moment.

The last option is the most desirable solution, but the most complex. Filters to hide information, and possibly landmarks (special nodes that always appear), will generally be needed. Maintaining the legibility of maps is the main purpose. The user will only visualize what is interesting to him, depending on his position (focus) in the hyperdocument structure and what is important regarding the overall structure of the hypermedia document. In order to build those filters, HyperProp uses an extension of the fisheye-view strategy for conceptual models that offer composite nodes [MSCS97 and MuSo96]. Another important feature of the fisheye-view strategy is that it allows users to choose the amount of information shown, tailoring the map according to their interests. Thus, users can choose to see more or less detail in the structural view. Another desirable characteristic is to be at two foci at once in the Structural View. This can be needed for example because some small structure is being copied from one place to somewhere else, or because a relation is being authored between two parts of the document.

Given a compound graph, there are several works proposing different types of layout. However a tool integrating filtering and graph layout design oriented to hypermedia users is still missing. Moreover, even the partial solutions already presented remains to be tested by a large number of naive users.

The second view, called Temporal View in HyperProp (Channel View in CMIFed jargon), is responsible for supporting the specification of temporal relationships among components of a hyperdocument, defining their relative position in time, as presented in the left plane of Figure 5. In this plane, nodes are represented by rectangles, whose lengths indicate the mean optimum duration of their presentation in a time axis and whose relative positions establish their temporal synchronization. A remarkable difference between the Temporal View and a timeline is that the time shown in the Temporal View is not explicitly specified by the author.

It is derived from temporal constraints, just as an approximation of the real-time presentation.

Temporal views are very useful to display partial time chains of objects, as will be specified in Section 3. However, when relations can be defined among selection and presentation events, as in more general models, it is difficult to represent the situation. This happens because a user interaction time instant is runtime dependent. HyperProp poor solution for this case is done by defining a region, called *limbo*, where nodes, that do not have their presentation time instant exactly determined, can be placed. Those nodes are dragged from the Structural View and dropped into the limbo. All events related to nodes in the limbo are assumed to have occurred in the past, but one cannot determine when. Placing objects in the limbo permits manipulating relations even though it is not known precisely when its source events have occurred. There is not yet an efficient graphical tool that permits the definition of non deterministic temporal relationships.

Temporal Views usually display the mean optimum duration of an object presentation. However, duration is usually specified as varying within an interval. In [KiSo95] the duration of an object is represented by an elastic time-box with a spring symbolizing the variation. There is not yet a tool that permits a good representation of time variations when several objects are involved.

Relations can be defined in the Temporal View, like links in NCM and *sync arcs* in CMIF. However, unless relations are stored in a separate repository, the composition that contains it should be defined using the Structural View. This is one of the reasons why these two views must work together.

Finally, the third view, called Spatial View in HyperProp (similar to Player in CMIFed jargon), supports the definition of spatial relationships among components of a document, establishing its presentation characteristics in a given device in a specific instant of time, as shown in the right plane of Figure 5. In this plane, rectangles represent spatial characteristics of objects, specifying, for example, their position in a video monitor or their volume level in an audio device, in a given specific time instant.

In CMIFed the player shows the effect of mapping the abstract document to a particular platform. It acts indeed as a document formatter (see Section 3). Sometimes, however, the author needs an abstract spatial view independent from the exhibition platform, like the Spatial View of HyperProp.

The Spatial View should allow spatial relation definition among its objects, like the Temporal View allows temporal relations definition. Obviously, spatio-temporal relations must be defined using the two views in an integrated way. Moreover, the relation may have to be placed inside a composition, so the Structural View must also be integrated to the two previous views.

Indeed, the three views should be related with each other. In NCM, the focused object in the Structural View is the basis for the time chain shown in the Temporal View. In the latter view, for each point in time, the Spatial View shows how components will be presented in the space defined by the output devices. Links and nodes defined in the Temporal View are immediately updated in the Structural View, and vice-versa. The same integration can be found in CMIFed.

The Spatial and Temporal Views can also have a very complex structure, requiring sophisticated algorithms to build the view. Filtering can also be needed in this case. An efficient design of these two views for a large number of nodes is a great problem to be faced.

Other open issues can be mentioned. There is not an efficient tool that guides author to define temporal and spatial consistent documents for a given particular platform. Indeed the complete design of a graphical authoring tool is an open issue. The few existent proposals are not definitive. They have to be tested for large documents authoring and with a large number of naive authors, in order that we can have feedback for better designs.

3 - Temporal and Spatial Formatting

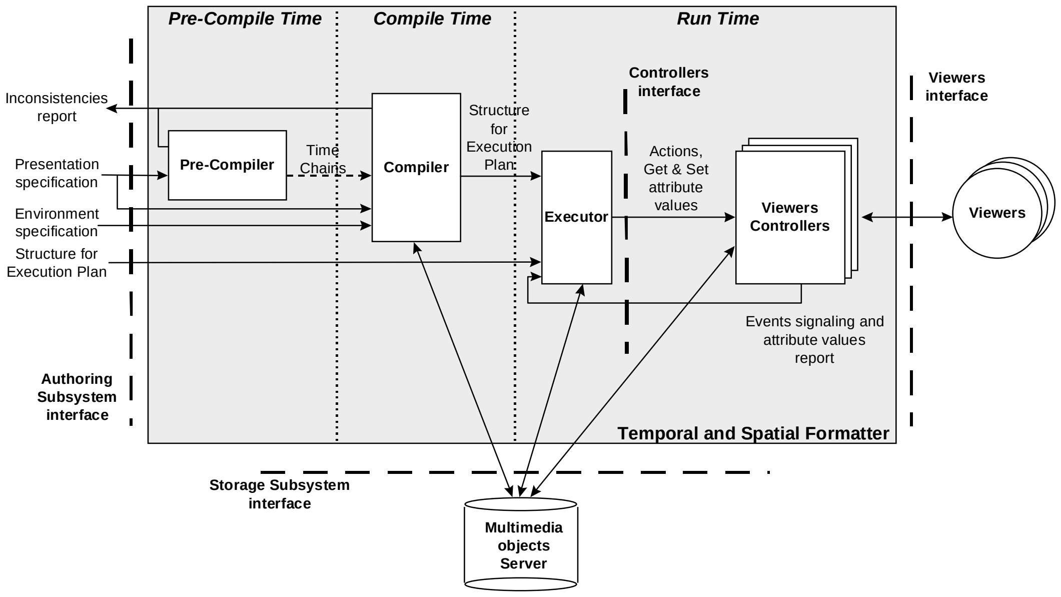
As mentioned hypermedia systems can be addressed in three different levels: storage, specification (authoring) and execution (formatting). Each one of these levels is based on a conceptual data model, that can be different from those of the other levels. In practice, choosing a single model for a particular application is not often sufficient. One needs both the flexibility of more abstract models and the accurate control over the presentation offered by more presentation-dependent models. As a result, one needs tools that can convert documents from a model into another.

Authoring models, as they are closer to users, usually are based on higher level abstractions. The great expressiveness of authoring models must be offered to users in an easy and comprehensible way. These are some of the reasons that led to the use of an event-driven model to specify the logical structure and presentation characteristics of a document. The input structure for the temporal and spatial formatter will be extracted from this model. As the formatter is closer to the operational machine, a lower level model for parallel state machines may be the most adequate for tasks like scheduling.

Almost all systems already mentioned in this paper follow different models for each one of its levels. For example, Firefly uses a time instant (event-driven) constraint based model during authoring, which is translated by a compiler (called *scheduler*) to partial time chains structured as timelines, which feed the runtime formatter. Firefly tries to avoid timeline problems by concatenating partial time chains to support unpredictable events. The compiler builds a main time chain (called *main temporal layout*) and other partial time chains (called *auxiliary temporal layout - time chains*), when a user asks for a document presentation. The formatter builds the execution plan from these structures. The CMIF formatter (called *Player*) has a compiler that generates, from the document specification, a directed graph of timing dependencies that feeds the presentation. The graph used by CMIF are similar to Petri nets. The I-HTSPN toolkit has a compiler (called *MHEG translator*) that converts the I-HTSPN specification (based on extended Petri net) into an MHEG representation. An MHEG machine should control the document presentation. HyperProp uses an object oriented time instant (event-driven) constraint based model (NCM) in the authoring phase, that is translated to an extension of TSPN (Time Stream Petri Net) [DiSe94] to model the execution plan. The extension simply consists of adding to outgoing arcs not only the minimum, optimum and maximum event duration, but also the expected duration calculated by a compiler and maintained by the HyperProp executor, as will be seen.

Other data models may coexist between authoring and execution phases. For example, in HyperProp, NCM can be converted to an RT-LOTOS specification for consistency checking as will be discussed later.

In order to address some issues on the storage and formatting phases of document handling we can use, just as an example, the formatter architecture of the HyperProp system, shown in Figure 6, that is very similar to that one of Firefly.



Pre-Compile Time
Compile Time
Run Time
Inconsistencies report
Presentation specification
Environment specification
Structure for Execution Plan
Pre-Compiler
Time Chains
Compiler
Structure for Execution Plan
Executor
Controllers interface
Actions, Get & Set attribute values
Viewers Controllers
Viewers interface
Viewers
Events signaling and attribute values report
Authoring Subsystem interface
Temporal and Spatial Formatter
Storage Subsystem interface
Multimedia objects Server

Figure 6 - HyperProp formatter architecture

The temporal and spatial formatter is responsible for controlling the document exhibition based on its presentation specification and the platform (or environment) description. The main idea is to build an execution plan (a schedule) to guide the formatter in its task. The execution plan should contain information about actions that must be fired when an event occurrence is signalized.

We say that an event is predictable when it is possible to know, a priori, its start and end relative time to another event. Otherwise, an event is called unpredictable. The sequence of event occurrences in time is called *time chain*. When the first event of a time chain is unpredictable, the time chain is called *partial time chain*. The execution plan is the set of all partial time chains of a document. It will guide the formatter in adjusting object duration on-the-fly, as well as in pre-fetching components' content in order to improve the presentation quality and to reduce the probability of temporal and spatial inconsistencies.

The HyperProp formatter architecture is composed by four elements: the pre-compiler, the compiler, the executor and the viewer controllers (or simply controllers), as shown in Figure 6.

Although the pre-compiler realizes formatting tasks, it is indeed a support to the authoring environment. Its main function is to check the temporal and spatial consistency of each document partial time chain. The pre-compilation is an incremental compilation done during authoring time. It can help the author to find errors, giving an instantaneous feedback whenever any inconsistency is detected, like mismatched time relationships (temporal inconsistency), or conflicts in a device use (spatial inconsistency) in an abstract ideal platform.

A document is called *intrinsically consistent* if no internal synchronization constraints can lead to inconsistent situations when the presentation is performed over an ideal platform. However, even if a document is intrinsically consistent, its presentation can lead to a deadlock situation, because the analysis is only based on the intrinsic (specified) duration of objects presentations. Therefore, we define a document as *extrinsically consistent* if its presentation in a real and resource-limited platform is also consistent. In this case, the verification is performed over the composition of the document specification and the platform specification, testing properties that are not intrinsic to the document, but depend on the behavior of its presentation over a specific platform. During pre-compilation time, only intrinsic consistency can be checked.

Considering the complete document presentation specification and the exhibition platform description, the compiler is responsible for generating the data structure used by the formatter to build the execution plan. At this time extrinsic consistency check can be done. Compilers can also realize intrinsic consistency checking.

As stated in several models, the event presentation duration can be specified as a tuple $< t_{min}, t_{opt}, t_{exp}, t_{max}, f_{cost}>$, where the minimum allowed duration (t_{min}), the maximum allowed duration (t_{max}), the duration that would give the best quality of presentation (t_{opt}) and the cost of shrinking or stretching the event duration (f_{cost}) are defined. The expected value (t_{exp}) is set initially to be equal to the optimum value. At compile time, the expected duration should be computed based on the cost minimization in order to warrant the spatial and temporal consistency of the document. This computation mechanism is called *elastic time computation* in [KiSo95].

The execution plan created by the compiler will guide the formatter in adjusting object duration on-the-fly, as well as in pre-fetching components' content in order to improve the presentation quality and to reduce the probability of temporal and spatial inconsistencies.

Based on the execution plan, the executor starts a controller and passes all information needed to create and control each object presentation. The executor may start one controller for all media objects or one controller for each media object, depending on the implementation. The controller creates the presentation of an object obtaining its corresponding data from the Multimedia Objects Server. From the viewers, controllers receive event signaling to be reported to the executor. The executor then updates the appropriate event state machine and evaluates all relations associated with this event. At the moment the controller reports an event to the executor, it can check the expected time in the execution plan to see if some adjustments are needed. Adjustments may be done through messages sent to controllers, telling them, for example, to accelerate the presentation rate, etc.

Note that the execution of a document can also be structured. Composition can invoke other compositions it contains. Each invoked composition returns control to its father once its execution has finished.

We can now present some open issues based on the reference architecture presented.

- **Media object pre-fetching**

Based on the execution plan the formatter can pre-fetch components' content in order to improve the presentation quality and to reduce the probability of temporal and spatial inconsistencies. In order to get the media object data content the formatter must negotiate the QoS it needs with the whole platform environment (operating system, network, hypermedia server, etc.). Unfortunately, QoS guarantees is still a broad open issue.

Based on QoS guarantees, if any is obtained, the formatter can estimate the worst delay to obtain a data content. This, however can be much larger than real delay, so the formatter can make use of other estimations. For example, CMIF initially estimates the delay based on heuristics involving data size; when a pre-fetch action is executed, the actual time it takes is saved and used as an estimate when the presentation is run again later.

When a document presentation has a lot of user interactions, pre-fetching is less effective. Again, when a presentation is played several times, statistics can be stored about which user-dependent relations are taken most often, to guide the selection of presentations to be firstly prepared.

Based on all the mentioned factors, heuristics should be developed to guarantee the QoS of a document presentation and minimize the elastic time adjustments necessary to maintain extrinsic consistency. Good heuristic proposals are welcome.

- **Object placement**

In order to guarantee the QoS of a presentation, a hypermedia server must distribute its objects in order to minimize its access delay and maximize its effective bandwidth. Some researches approach the problem focusing in a specific matter. For example, [GRAQ91] proposes an object placement based on its probabilities of access and size. The problem is how to estimate such probabilities. Note that heuristics to estimate these probabilities are closely related with those ones used in the pre-fetching. As another example, [VaTS98] proposes an indexing scheme based on multi-dimensional (spatial) data structures for efficient handling of queries related to spatio-temporal relationships. Other database works discuss the efficient storage of versions of the same object.

Indeed, the object placement must involve all issues related to versioning, cooperative work, spatio-temporal synchronization, document access frequency, etc. A general algorithm or heuristic is thus necessary and is still an open issue.

Also the definition of a complete query language that allows not only content queries but also structure queries are necessary. Again, there are some proposals for specific topics, for example, for querying version structures, querying spatio-temporal structures and querying context structures. Structure query is closely related to the object placement issue and also needs a solution. Note also that structure queries are closely related with the automatic generation of documents mentioned in Section 2.1.

- **Elastic time computation**

In order to guarantee document consistency, the expected duration of an object presentation should be computed based on cost minimization. The elastic time computation may be realized during compile time. Some possible algorithms are discussed in [BuZe93 and KiSo95]. The problem is however much more complex when this computation must be in real time. This can happen when the executor has to shrink or stretch an object presentation in order to compensate for some unpredictable platform delay. When this is done, the expected duration for remaining objects to be presented should be computed again, and in real time. That is the reason why the few systems that implement elastic time adjustment only do it at compile time (Firefly and NCM).

- **Consistency checking**

In spite of all facilities offered by hypermedia authoring tools, few attention has been paid to the design of documents free from inconsistencies that come from temporal and spatial constraints. In general, inconsistencies come from several combined spatio-temporal constraints that depend on the presentation environment. Detecting them is a hard task, claiming for a design validation methodology. The benefits of formal methods for multimedia and hypermedia document validation have been exposed in several papers [CoOl95, CoOl96, WSSD96 and JLRS97]. Thus consistency checking usually includes an automatic translation from a high-level hypermedia model into an FDT (formal description technique), since authoring using an FDT can be very tedious an difficult to naive users. The FDT is therefore completely hidden to authors. Having an FDT document specification, general-purpose validation techniques (like reachability analysis, model-checking) can be applied for analyzing application oriented properties of the design, in particular, consistency properties.

How to perform such mapping is a question to be answered. The selected formal method should present some unique features in order to be useful as well as efficient:

- It must have a formal semantics (and not only an intuitive semantics).
- It should emphasize composition, in the sense that complex document structures should be expressed not by ad-hoc constructions.
- It should be able to express non-deterministic behaviors, in particular interactions with the external environment (e.g., the users).
- It should have the ability to express complex temporal-constraint behaviors.
- Finally, it should be mature enough, and software tools for automating the validation procedure should be available.

Some works in the area use process algebra as a target FDT. Process algebra meets the first three requirements and LOTOS (Language of Temporal Ordering Specification) becomes a strong candidate for being an international standard. Since standard LOTOS is not able to express temporal-constraint behaviors, different extension proposals have been made within the international LOTOS community, and are currently being standardized. Among these proposals, RT-LOTOS (Real-Time LOTOS) was chosen by [SSSC98] for the NCM formal specification. In particular, because the RTL (Real-Time LOTOS Laboratory) software tool is available and operational. The same approach might easily be adapted to the new emerging LOTOS standard (E-LOTOS) once stabilized with an adequate tool support.

Another example comes from the I-HTSPN toolkit that has a pre-compiler (called Analyzer) that checks temporal and spatial inconsistencies of multimedia modeled scenarios using time Petri nets.

Consistency checking is not an easy task and is closely related to elastic time computation issue. Only the first steps were given in its direction.

4 - OHS and the WWW

Most recently, research emphasis on OHSs (Open Hypermedia Systems) has changed from the system conception itself to its integration with other applications. The effort to develop a standard, open and powerful hypermedia system adopted world wide has changed little by little to the integration with other world wide *de facto* accepted applications. Obviously, the main integration target has been the World-Wide Web, as it is the largest distributed hypermedia system in use. The integration of OHSs and the WWW is motivated by trying to solve WWW's limitations, using more sophisticated and powerful hypermedia models, while exploring the Web large distribution and standards.

Undoubtedly, one of the main reasons for the WWW popularity is its simplicity and easy-of-use. However, the WWW has some limitations as a hypermedia system, such as:

- its data model defines links embedded in nodes (HTML pages), resulting in some shortcomings:
  - it does not allow separation between referenced data and references (links) itself, what makes link and data maintenance, and data reuse without inheriting relations, difficult;
  - it does not permit creating references in pages where write access is not granted;
  - it requires a special content format (e.g., HTML, VRML);
  - links can just be followed in one direction, avoiding us to know which links reference a certain document.
- one can only define unidirectional point-to-point "go to" relations (1:1 links) and there is no support for defining temporal and spatial synchronization relationships;
- standard characteristics of open hypermedia systems such as guided tours and structural views of documents are not offered;
- there is no support for version control and cooperative work.

The integration of OHSs and the WWW is motivated by trying to solve these limitations, while exploring the Web large distribution and standards. The main OHS/WWW integration purposes can be resumed as:

1. incorporate OHS facilities to WWW documents, allowing definition of compositions, navigation in structural-view browsers, creation of links touching WWW nodes independent of having write access to them, creation of bi-directional m:n links, definition of temporal and spatial synchronization relationships, definition of guided tours, support for version control and cooperative work;
2. use WWW browsers to present OHS documents;
3. allow OHS documents to reference WWW documents and vice-versa.

There are many proposals for integrating OHSs and the WWW, from which we can highlight Chimera [Ande97], Hyper-G [AnKM95], Microcosm [CRHH95, HaDH96], DHM [GrBS97], and NCM/WWW [RoMS98]. They can be classified in: compile time integration [HaDH96] and runtime integration [Ande97, AnKM95, CRHH95, GrBS97, RoMS98]. In compile time integration, as name suggests, a total translation of documents from one hypermedia model to the other is made before presentation, while in runtime integration, this conversion is done progressively, during user navigation. Compile time integration almost always limits OHS's potentialities. It requires the development of translation tools, but does not require changes in the systems architectures. On the other hand, runtime integration requires a new architecture that puts together basic elements from both systems architectures (servers, clients, protocols, data formats, etc.). Some possibilities for combination of these basic elements are discussed in [Ande97].

Microcosm has developed tools for compile time integration. The solution has, however, many restrictions, as discussed in [Ande97]. Microcosm also proposed the Distributed Link Service, DLS, for WWW users. DLS allows clients to connect to link servers and request a set of links to be applied on data in WWW documents. The system permits users to subscribe to many different linkbases. There is a main link database for the server, which is always used, and additional link databases from which the user may choose. These additional databases allow the server to offer a range of different link sets, known as contexts. However, as there is not a notion of nested composition, it is impossible to build a new linkbase inheriting relations defined in another one. In other words, users must know all linkbases that compose the context. Moreover, the lack of compositions does not allow structuring WWW documents.

Some integration experiments were developed by Chimera. This OHS allows organizing WWW nodes hierarchically in its hyperwebs, but it stores links in separate databases, permitting links between hyperwebs in its current version. Although the notion of compositions is embedded in the hyperwebs, they only group nodes, and the system does not have the possibility of relation (link) inheritance.

Hyper-G is a large-scale distributed hypermedia system and one of the first OHSs to integrate with the Web. It also permits nested compositions of nodes, but, like Chimera, it stores links in a separate database, also not allowing relation (link) inheritance.

The HyperDisco [WiLe97] approach uses the concept of workspaces as a collection of nodes and links, offering a wide range of hypermedia services to applications. However, in its current implementation, not all facilities of HyperDisco workspaces are already implemented. Indeed, just a few of them are implemented and the others are addressed as future works. Link inheritance and nested composite nodes are addressed as future work.

In the current implementation of NCM integration [RoSo98], not all NCM facilities are yet incorporated in WWW documents. In short, among integrated facilities are: separation of links from node’s content, definition of compositions to structure documents (allowing link inheritance and reuse of structures of nodes and links), visualization of documents structure, user navigation through graphical views, and navigation through guided tours and trails that maintain the document navigation history. Support for version control, cooperative work and the definition of n-ary relations that could also specify spatial and temporal synchronization among nodes are addressed as future works, in progress.

Integration experiments are also related in DHM/WWW integrated system. DHM conceptual model separates links from nodes’ content, but it does not offer nested compositions and it stores links in a unique database, having the same shortcomings mentioned for the Chimera solution.

DLS client interface was designed to support the use of Netscape for communication with DLS servers. The nature of the client interface and the way it communicates depend on the particular platform being used. Hyper-G requires a special browser (and also a special text document format, HTF) in order for users to achieve all benefits of the system, including the ability to create links and collections. DHM and NCM/WWW present a platform-independent solution, also based in the Netscape browser using Java applets, JavaScript and

LiveConnect. DHM also presents two other platform-dependent solutions, one using Netscape plugins and other using Internet Explorer.

None of the related works offer a graphical tool for helping authoring and user navigation. In the near version, NCM/WWW promises to incorporated the Structural View of HyperProp.

Hard work must be done in OHS/WWW integration regarding spatio-temporal synchronization. Except for NCM integration, none of the systems mention mechanisms for temporal and spatial synchronization relationships, probably because their conceptual data models do not handle them. NCM has a well-defined support for synchronization relationships that will also be incorporated in future versions of NCM/WWW using the present NCM formatter, fully implemented as a Java object. However, a lot of questions must be answered before. For example: How can we map a synchronization sequence of objects on HTML pages? What is the corresponding entity for a page in a event driven OHS model? How do we handle contextual links? For example, if the relation specifies that a source context for a link must remain, we must open another browser to show the destination context or show this context in the same page which maintains the source context, for example in another HTML frame? How can we use style sheets for synchronization?

Another problem to be faced is to implement a platform and browser independent solution for the OHS and WWW integration. Unfortunately, this is not possible yet, as evidenced by all works related to WWW integration. However, a natural tendency is that WWW browsers become more opened, and Java language offers more features, allowing a more general solution in a near future. We need more proposals and experimental works to contribute in this direction. Maybe, one of the proposals may have its facilities incorporated to Web browsers, as suggested by SMIL.

5- Final Remarks

In spite of the number of works related to multimedia and hypermedia authoring and formatting, and in spite of advances already obtained, a long and winding road still remains to be trek. This paper considered some issues on the development of multimedia and hypermedia authoring and formatting tools, examined the current state of the art, and then discussed a set of research challenges that need to be addressed.

The goal was not to present a complete work on open issues on document authoring and formatting, but only to mention some of them and to stress that some are old issues [Hala88] that still remains, although now with different taste given by partial results already obtained. We hope to have contributed in this direction.

References

[Ande97] Anderson, K.M. “Integrating Open Hypermedia Systems with the World Wide Web”. *The Eighth ACM International Hypertext Conference*. Southampton, UK, April 1997. pp.157-166.

[AnKM95] Andrews, K.; Kappe, F.; Maurer, H. “Serving Information to the Web with Hyper-G”. *Computer Networks and ISDN Systems 27 (6)*, 1995. pp. 919-926.

[BuHa95] Bulterman, D.C.A.; Hardman, L. “Multimedia Authoring Tools: State of the Art and Research Challenges”. *Computer science today: recent trends and developments*, edited by Jan van Leeuwen, Springer Lecture Notes in Computer Science 1000, 1995, pp. 575-591.

[BuZe93] Buchanan, M.C.; Zellweger, P.T. “Automatic Temporal Layout Mechanisms”. *Proceedings of ACM Multimedia’93*, California, 1993. pp. 341-350.

[CoOl95] Courtiat, J.P; Oliveira, R.C. “On RT-LOTOS and its application to the formal design of multimedia protocols”. *Annales des Télécommunications*, no 11-12. France. December 1995.

[CoOl96] Courtiat, J.P.; Oliveira, R.C. “Proving Temporal Consistency in a New Multimedia Synchronization Model”, *Proc. of the ACM Multimedia’96*, November 1996.

[CRHH95] Carr, L.; Roure, D.; Hall, W.; Hill, G. “The Distributed Link Service: A Tool for Publishers, Authors and Readers”. *Fourth International World Wide Web Conference*, Boston, USA. 1995.

[CTRS91] Casanova, M.A.; Tucherman, L.; Lima, M.; Rodriguez, N.R.; Soares, L.F.G. “The Nested Context Model for Hyperdocuments”. *Proceedings of Hypertext’91*. Texas. December 1991.

[DiSe94] Diaz, M.; Sénac, P. “Time Stream Petri Nets, a Model for Timed Multimedia Information”. *Proceedings of the 15th International Conference. on Application and Theory of Petri Nets*, Zaragoza, 1994. pp. 219-238.

[GRAQ91] Ghandeharizadeh, S.; Ramos, L.; Asad, Z.; Qureshi, W. “Object Placement in Parallel Hypermedia Systems”. *Proceedings of Hypertext ' 91*Texas. 1991.

[GrBS97] Grønbœk, K.; Bouvin, N.O.; Sloth, L. “Designing Dexter-based hypermedia services for the World Wide Web”. *The Eighth ACM International Hypertext Conference*. Southampton, UK, April 1997. pp.146-156.

[Haak92] Haake, A. “Cover: A Contextual Version Server for Hypertext Applications”. *Procceedings of European Conference on Hypertext, ECHT' 92*Milano. December 1992.

[Haak94] Haake, A. “Under CoVer: The Implementation of a Contextual Version Server for Hypertext Applications”. *Proceedings of the European Conference on Hypertext*. Edimburgo. Setembro de 1994.

[Haak96] Haake, A.; Hicks, D. “VerSe: Towards Hypertext Versioning Styles”. *Proceedings of Hypertext 96*. Washington, EUA. Setembro de 1996.

[HaBR93] Hardman, L.; Bulterman, D.C.A.; van Rossum, G. “The Amsterdam Hypermedia Model: extending hypertext to support *real* multimedia”. *Hypermedia Journal*, Vol. 5(1), July 1993. pp. 47-69.

[HaBR94] Hardman, L.; Bulterman, D.C.A.; van Rossum, G. “The Amsterdam Hypermedia Model: Adding Time and Context to the Dexter Model”. *Communications of the* ACM, 37 (2), February 1994, pp. 50-62.

[HaBu95] Hardman, L.; Bulterman, D.C.A. “Towards the Generation of Hypermedia Structure”. *First International Workshop on Intelligence and Multimodality in Multimedia Interfaces*, Edinburgh, UK, July1995.

[HaDH96] Hall, W.; Davis, H.; Hutchings, G. “Rethinking Hypermedia: The Microcosm Approach”. *Kluwer Academic Publishers, Norwell*, Massachusetts, USA, 1996.

[Hala88] Halasz, F.G. “Reflections on NoteCards: Seven Issues for the Next Generation of Hypermedia Systems”. *Communications of the ACM*, Vol. 31, No. 7, 1988, pp. 836-852.

[HaWB98] Hardman, L.; Worring, M.; Bulterman, D.C.A. “Integrating the Amsterdam Hypermedia Model with the Standard Reference Model for Intelligent Multimedia Presentation Systems”. *Computer Standards and Interfaces*, vol. 18, 1998.

[ISO96] ISO/IEC. “Information Technology - Processing languages - Document Style Semantics and Specification Language (DSSSL)”. *International Standard ISO/IEC 10179:1996*, 1996.

[JLRS97] Jourdan, M.; Ladayia, N.; Roisin, C.; Sabry-Ismail, L. “An Integrated Authoring and Presentation Environment for Interactive Multimedia Documents”. *4th Int. Conf. on MultiMedia Modeling — MMM’97*, Singapore, Nov. 1997. pp. 247-262.

[KiSo95] Kim, M.Y.; Song J. “Multimedia Documents with Elastic Time”, *ACM Multimedia’95*, San Francisco, California, November 1995.

[LiBo96] Lie, H. W.; Bos, B. “Cascading Style Sheets. level 1”, *W3C Proposal Recommendation*, 1996. *http://www.w3.org/pub/WWW/TR/.*

[MSCS97] Muchaluat, D.C.; Soares, L.F.G.; Costa, F.R.; Souza, G.L. “Graphical Structured-Editing of Multimedia Documents with Temporal and Spatial Constraints”. *Proceedings of the Multimedia Modeling - MMM' 97*, Singapore, November 1997, pp. 279-295.

[Muns96] Munson, E. “A New Presentation Language for Structured Documents”. *Proceedings of the International Conference on Eletronic Documents, Document Manipulation and Document Dissemination - EP96*, Palo alto, September 1996.

[MuSo96] Muchaluat, D.C.; Soares, L.F.G. “Fisheye Views for Compound Graphs”. *Technical Report of Laboratório TeleMídia - PUC-Rio*, Rio de Janeiro, Brasil, 1996. *Also submitted to Graph Drawing’98, Montreal, Canada, 1998.*

[NaKa97] Nang, J.; Kang, S. “A New Multimedia Synchronization Specification Method for Temporal and Spatial Events”. *Proceedings of IEEE International Conference on Multimedia Computing and Systems - ICMCS’97*, Ottawa, Canada, June 1997. pp. 236-243.

[Oste92] Osterbye, K. “Structural and Cognitive Problems in Providing Version Control for Hypertext”. *Procceedings of European Conference on Hypertext, ECHT' 92*, Milano. December 1992.

[PeLi96] Pérez-Luque, M.J.; Little, T.D.C. “A Temporal Reference Framework for Multimedia Synchronization”. *IEEE Journal on Selected Areas in Communications (Special Issue: Synchronization Issues in Multimedia Communication*), Vol. 14, No. 1, January 1996, pp. 36-51.

[RJMB93] van Rossum, G.; Jansen, J.; Mullender, K.S.; Bulterman, D. “CMIFed: A Presentation Environment for Portable Hypermedia Documents”. *Proceedings of ACM Multimedia’93*, California, 1993. pp. 183-188.

[RoSo98] Rodrigues, R.F.; Soares, L.F.G. “Composite Nodes and Links on the World-Wide Web”. *Technical Report of Laboratório TeleMídia - PUC-Rio*, Brasil, March 1998. *Also in IV Brazilian Conference on Multimedia and Hypermedia Systems - SBMIDIA’98. Rio de Janeiro, Brasil, May 1998.*

[SaBr94] Sarkar, M.; Brown, M. H. “Graphical Fisheye Views”. *Communications of the ACM*, Vol. 37 No. 12, December 1994.

[Soar98] Soares, L.F.G. “HyperProp - An Open Hypermedia System” *Proceedings of the Workshop of ProTeM-CC.* Belo Horizonte, Brazil. April 1998. pp.203-238.

[SoCR94] Soares, L.F.G.; Casanova, M.A.; Rodriguez, N.R. “Nested Composite Nodes and Version Control in Hypermedia Systems”. *Proceedings of the Workshop on Versioning in Hypertext Systems*, in connection with ACM European Conference on Hypermedia Technology, Edinburgh. September 1994.

[SoCR95] Soares, L.F.G.; Casanova, M.A.; Rodriguez, N.L.R. “Nested Composite Nodes and Version Control in an Open Hypermedia System”. *International Journal on Information Systems; Special Issue on Multimedia Information Systems*, September 1995. pp. 501-519.

[SSSC98] Santos, C.A.S.; Soares, L.F.G.; Souza, G.L.; Courtiat, J.P. “Design Methodology and Formal Validation of Hypermedia Documents”. To appear in *Proceedings of ACM Multimedia’98*. Bristol, UK. September 1998.

[VaMo93] Vazirgiannis, M. Mourlas C. “An Object Oriented Model for Interactive Multimedia Applications”. *The Computer Journal, British Computer Society*, vol. 36 (1), January 1993.

[VaTS96] Vazirgiannis, M.; Theodoridis, Y.; Sellis, T. "Spatio - Temporal Composition in Multimedia Applications". *Proceedings of the International Workshop on Multimedia Software Development, IEEE-ICSE '96* - Berlin, 1996.

[VaTS98] Vazirgiannis, M.; Theodoridis, Y.; Sellis, T. "Spatio-Temporal Composition and Indexing for Large Multimedia Applications". *to appear in ACM / Springer Multimedia Systems, Springer - Verlag,* Vol 6(5), September 1998.

[Vazi96] Vazirgiannis, M. "Multimedia Data Base Object and Application Modelling Issies and an Object Oriented Model". *in the book "Multimedia Database Systems: Design and Implementation Strategies" (editors Kingsley C. Nwosu, Bhavani Thuraisingham and P. Bruce Berra), Kluwer Academic Publishers*, 1996, pp 208-250.

[VSTH98] Vazirgiannis, M.; Stamati, I.; Trafalis, M.; Hatzopoulos, M. "Interactive Multimedia Scenario: Modeling & Rendering". *to appear in the proceedings of Multimedia track of ACM - SAC'98 Conference,* 1998.

[W3C98] Synchronized Multimedia Work Group of the World Wide Web Consortium, "Synchronized Multimedia Integration Language (SMIL) 1.0 Specification". *W3C Proposed Recommendation*. 1998. http://www.w3.org/Press/1998/SMIL-PR

[WBHT97] Worring, M.; Berg, C.; Hardman, L.; Tam, A. "System Design for Structured Hypermedia Generation". *LNCS 1306, Visual Information Systems, ed. C. Leung*, 1997.

[WiLe97] Wiil, U.; Leggett, J. "Workspaces: The HyperDisco Approach to Internet Distribution". The Eighth ACM International Hypertext Conference. Southampton, UK, April 1997. pp.146-156.

[WSSD96] Willrich, R.; Saqui-Sannes, P.; Sénac, P.; Diaz, M. "Hypermedia Document Design Using the HTSPN Model". *Third International Conference on MultiMedia Modeling — MMM'96,* Tolouse, November 1996. pp. 151-166.